Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) An execution unit for execution of multiple context threads comprises:

an arithmetic logic unit to process data for executing threads;

control logic to control the operation of the arithmetic logic unit;

a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set constructed with a two-ported random access memory architecture, with the register set divided into a plurality of banks, each bank having no more than two ports and being capable of performing a read and a write to two different words with the two ports in the same processor cycle.

2. (Original) The execution unit of claim 1 wherein the register set is logically partitioned into a plurality of relatively addressable windows.

3. (Original) The execution unit of claim 2 wherein the number of windows of the register set is according to the number of threads that can execute in the processor.

4. (Original) The execution unit of claim 1 where the relative addressing allows the currently executing thread to

access to any of the registers relative to the starting point of a window of registers.

5. (Original) The execution unit of claim 1 wherein the register set is absolutely addressable where any one of the addressable registers may be accessed by the currently executing thread by providing the exact address of the register.

6. (Original) The execution unit of claim 1 wherein the control logic further comprises:

context event switching logic fed by signals from a plurality of shared resources with the signals causing the context event logic to indicate that threads are either available or unavailable for execution.

7. (Original) The execution unit of claim 6 wherein the control logic addresses a set of memory locations for storing a list of available threads that correspond to threads that are ready to be executed and a set of memory locations for storing a list of unavailable threads that are not ready to be executed.

Claims 8-14 (Canceled)

15. (Currently amended) A method for executing multiple context threads comprises:

processing data for executing threads within an arithmetic logic unit;

operating control logic to control the arithmetic logic unit;

storing and obtaining operands for the arithmetic logic unit within a general purpose register with the register set

constructed with a two-ported random access memory architecture; and

arranging the register set into a plurality of banks, each bank having no more than two ports and being capable of performing a read and a write to two different words with the two ports in the same processor cycle.

16. (Original) The method of claim 15 wherein the register is relatively addressable.

17. (Original) The method of claim 15 further comprising:

arranging the register set into a number of windows according to the number of threads that can execute in the processor.

18. (Original) The method of claim 17 wherein storing and obtaining further comprises:

addressing the registers by the currently executing thread by providing a relative register address that is relative to the starting point of a window of registers.

19. (Currently amended) A processor unit comprises:

an execution unit for execution of multiple context threads comprising:

an arithmetic logic unit to process data for executing threads;

control logic to control the operation of the arithmetic logic unit;

a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set constructed with a two-ported random access memory architecture, the register set

being arranged into a plurality of banks, each bank having no more than two ports.

20. (Original) The processor of claim 19 wherein the register set is logically partitioned into a plurality of relatively addressable windows where the number of windows of the register set is according to the number of threads that can execute in the processor.

21. (Original) The processor of claim 20 where the relative addressing allows the currently executing thread to access to any of the registers relative to the starting point of a window of registers.

22. (Original) The processor of claim 20 wherein the register set is absolutely addressable where any one of the addressable registers may be accessed by the currently executing thread by providing the exact address of the register.

23. (Original) The processor of claim 20 further comprises:

a set of memory locations for storing a list of available threads that correspond to threads that are ready to be executed;

a set of memory locations for storing a list of unavailable threads that are not ready to be executed; and

context event switching logic fed by signals from a plurality of shared resources with the signals causing the context event logic to indicate which threads are either available or unavailable for execution.

24. (Original) The processor of claim 23 wherein execution of a context swap instruction causes a currently running thread

to be swapped out to the unavailable thread memory set and a thread from the available thread memory set to begin execution within a single execution cycle.

25. (Original) The processor of claim 23 wherein execution of a context swap instruction specifies one of the signal inputs and upon receipt of the specified signal input causes the swapped out thread to be stored in the available thread memory set.

26. (Original) The processor of claim 23 wherein execution of a context swap instruction specifies a defer_one operation which causes execution of one more instruction and then causes the current context to be swapped out.

Claims 27-29 (Canceled)